Claremont McKenna College

Predicting Outcomes of El Clásico Using Random Forests and Extreme Gradient Boosting

submitted to
Mark Huber

by
Emanuel Jarquin

for
Senior Thesis
Spring 2022
April 25, 2022

Abstract

In the modern era, sports betting is becoming increasingly popular. This is especially true in the realm of soccer (or 'football' as it is known outside the United States). As a result, the concept of attempting to predict the outcomes of soccer matches using machine learning has garnered much attention in recent years. In this thesis, I utilize well-known machine learning techniques to predict the outcomes of *El Clásico* matchups and compare the predictive performance of these techniques. The predictive methods employed for this thesis are random forests using the `party` package in `R` and extreme gradient boosting using the `xgboost` package. The dataset that will be used has been created using historical soccer data that includes match and team statistics.*

Table of Contents

* The code and data used for this thesis can be accessed at https://github.com/ejarquin22/Senior-Thesis/

1 Introduction

1.1 Soccer and Sports Betting

The world of sports has been a staple in human activity throughout history. Sports are a notion that has always been mainstream and has continued to gain popularity as sports events have continued to evolve. This is particularly true for soccer. Soccer is a sport in which there are two teams, each made up of 11 players, attempting to score goals by getting the ball into the opposing team's goal. Whichever side scores the most goals at the end of two 45-minute halves wins. However, if both teams score the same number of goals at the end of the 90 minutes of regulation time, then the game ends in a draw. Additionally, both teams must try and score a goal by maneuvering the ball without using their hands. Although the history of soccer and how the sport originated is unclear, the history of the modern game of soccer and its origin are tied directly to 19th century Britain. [1]

Since its origin, the modern game of soccer has seen a constant rise in popularity around the world. Concurrently, the number of people who bet on soccer matches has also witnessed a constant rise in popularity. In fact, the number of people who participated in sports betting nearly doubled in the United States in 2021 alone. [2] Logically, being able to predict the outcome of a soccer match could give anyone an edge in sports betting. Unsurprisingly, the subject of match predicting has also grown in conjunction with the popularity of soccer match betting. This has been escalated through the progression in the development of machine learning techniques whose main purpose is to learn how to form predictions.

1.2 El Clásico

Founded in 1902 and representing the capital of Spain, Real Madrid Club de Fútbol (Real Madrid) is arguably the most successful soccer team of all time at the club level. [3] Having won a record 33 domestic league titles, as well as a record 13 European championship titles, Real Madrid has earned itself the honor of being one of the most decorated clubs in the history of modern soccer. Fútbol Club Barcelona (Barcelona) represents the Spanish city of Barcelona and was founded in 1899. Barcelona is perhaps the second most successful soccer club in the history of modern soccer. Since Barcelona was founded, the club has won 26 domestic league titles, in addition to their 5 European championships [4]. Needless to say, both clubs are home to two of the largest fan bases in the realm of club soccer.

El Clásico is the matchup between the two soccer giants, Real Madrid and Barcelona. El Clasico occurs at least twice every year as part of Spain's domestic league, La Liga. However, every year there is the possibility for the two sides to match up against one another in the European Champions League or either of Spain's two domestic

tournaments, the Spanish Super Cup and the Copa del Rey. This matchup is considered to be one of the biggest soccer matchups in the world due to the fact that both teams are two of the most successful soccer clubs ever. Additionally, both sides are usually capable of showcasing some of the greatest soccer performances that one could ever witness in a game of soccer. This is especially true when both teams play against one another. On account of El Clásico being so popular, it is not surprising that the matchup attracts the attention of a myriad of soccer bettors from around the world every time it occurs. Consequently, analyzing El Clásico matchups and learning to predict the outcome of these matches could benefit the progression of the subject area of predicting soccer matches using machine learning.

1.3 Relevant Work

Schauberger and Groll (2018) have influenced this thesis through the work they established by comparing different machine learning techniques using data from the 2002-2014 FIFA World Cups. [5] Their findings indicate that random forests models slightly outperform regression-based models from a predictive standpoint. The random forest model built by Schauberger and Groll (2018) and their list of dataset features has served as a framework for the development of the random forests model developed for this thesis. This thesis aims to compare the predictive performance of random forests to another machine learning technique, extreme gradient boosting (xgboost).

The next section of this thesis will outline the process in which the dataset that will be used for the analysis of the two predictive models was constructed. Section 3 will illustrate some of the difficulties faced throughout the modeling phase and will delineate both the random forests and xgboost machine learning techniques. Finally, Section 4 compares the results between the two models and questions how the results of each model change with respect to the dataset.

2 Data

2.1 Building the Dataset

To compare the predictive performance between random forests and xgboost, the models must be trained and tested using some form of data. For this thesis, the utilized dataset is made up of data from every competitive El Clásico matchup from the 2008-2009 season up until the 2021-2022 season. This dataset includes some features used by Schauberger and Groll (2018) in conjunction with additional features that represent in-game statistics from each match-up. The dataset is made up of 45 observations, each of which accounts for an El Clásico matchup. For every observation, there are 34 features, which include in-game match statistics, team statistics, and some economic factors for each side. All the features in this dataset are represented either as

numeric or character data types. Here is a short description of each feature that is present for each El Clásico matchup that is a part of the dataset.

- **Division:** The competition that the matchup took place in (domestic league, European tournaments, or domestic tournaments).
- **Date:** The date that the matchup occurred.
- **HomeTeam/AwayTeam:** Indicates the side was playing at home and the side was playing away from home.
- **FTHG/FTAG:** The full-time goal counts for the home and away teams respectively.
- **HTHG/HTAG:** The half-time goal counts for the home and away teams respectively.
- **FTR:** The result of the match at full-time. Represented as H: home win, D: draw, and A: away win.
- **HTR:** The result of the match at half-time. Represented as H: home win, D: draw, and A: away win.
- **HS/AS:** The number of shots taken by the home and away teams respectively.
- **HST/AST:** The number of shots that were on target from the home and away teams respectively.
- **HF/AF:** The number of fouls committed by the home and away teams respectively.
- **HY/AY:** The number of yellow cards accumulated by the home and away teams respectively.
- **HR/AR:** The number of red cards accumulated by the home and away teams respectively.
- **365H/365D/365A:** The betting odds for each possible match outcome; a home win, a draw, and an away win.
- **Age:** The average age of each squad.
- **Population:** The population of the city from which each side is based.
- **GDP:** The GDP per capita as a ratio of each side's respective city and the country of Spain.
- **HomeRank/AwayRank:** The rankings of each team in their domestic league at the time of each matchup.
- **HC. Tenure/AC. Tenure:** The length of tenure for the home and away team managers.
- **HC. Nationality/AC. Nationality:** Indicates whether each of the managers is of the same nationality as the club.
- **HC.Age/AC.Age:** The average age of the home and away team managers.

Assembling the complete dataset was a lengthy process due to the fact that a dataset that included all El Clásico matches, the team details, and match statistics for each matchup was not readily available at the start of this thesis. Consequently, the dataset would have to be built manually. Much time was spent searching for a dataset that at least

contained all El Clásico matchups and the result of each matchup, however, this endeavor turned out to be successful. It seemed as though the entire dataset would have to be built from scratch until a webpage that contained a list of datasets for each season in La Liga's history was found. [6] Each of these datasets contained all the matchups that occurred within a single La Liga season. After combining all the separate datasets for each season into one single dataset, the El Clásico matchups that occurred in Spain's domestic league could be separated into another data frame through filtering. This resulted in the initial version of the dataset that would go on to be used for the analysis of the random forests and xgboost techniques.

This initial dataset was a solid foundation to build upon, however, there were still many pieces missing from what would become the final dataset. Firstly, this dataset only contained El Clásico matchups that occurred in La Liga. Unfortunately, another dataset that was similar to the initial dataset for the matchups that occurred outside of La Liga was unavailable. This meant that the matchups between Real Madrid and Barcelona that happened outside of their domestic league would have to be incorporated into the dataset manually. Apart from only containing El Clásico matchups that occurred in La Liga, the initial data set did not include many of the features that would appear in the final dataset. These features include age, population, GDP, rank, and information about the coach for each side of each matchup. The incorporation of these features into the dataset also would have to be done manually. Although both of these processes took a lot of time to complete, incorporating the additional matchups proved to be far more difficult than incorporating the additional features. What made incorporating the additional matchups so difficult was that there did not exist a single soccer match statistics site that contained all the necessary stats from each El Clásico matchup. Stats like the number of fouls committed by each side, the number of shots taken by each side that were on target, and the number of cards each side accumulated never consistently showed up together with the other match statistics. So, for each additional El Clásico matchup that occurred outside of La Liga, stats from different soccer match statistic webpages had to be used. [7] [8] Additionally, some statistics for El Clásico matchups from the earlier seasons (2008-2013) were unavailable on those pages. This led to the use of even more soccer match statistic websites in order to find the missing information. [9] [10] Regardless of the trouble encountered while building upon the initial dataset, what was left was a final dataset that was clean and ready to be used for the analysis of the two predictive models.

3 Predictive Models

3.1 Initial Challenges

There were several challenges faced throughout the process of setting up the two predictive models. The most notable of which had to do with the datatypes of the feature columns in the dataset. The issue of the data types arose when it came time to train the random forests model. The problem was that the random forests function `cforest`,

from the `party` package in `R`, does not work if any of the feature columns in the training or testing set are of `character` data type. To circumvent this problem, the data types of all the feature columns in the dataset had to be changed to be type `factor`, which can be done using the `lapply` function from `R`'s `base` package. This seemed to be the end of the data type problems at the time, however, when it came time to train the xgboost model, a similar problem came to light. The problem now was that the `xgboost` function does not work with datasets that include feature columns that are of `factor` or `character` data type. To resolve this problem, a copy-dataset was created from the dataset being used to train and test the random forests model. Then, the data types from each feature column in the copy-dataset were reverted from being a `factor` back to the original data type. Additionally, to solve the issue, we would have to drop all of the feature columns that were of type `character`. This included Date, Division, HomeTeam, AwayTeam, FTR, and HTR. Dropping Date, Division, HomeTeam, and AwayTeam would not be a problem since these features are details about each matchup rather than measures that occurred as a result of the matchup. Therefore, these features were not going to be useful predictors anyways. However, having to drop FTR and HTR from the training and testing sets would have been a major issue. On one hand, FTR is used as the response variable for both models, so leaving this feature column out of the dataset would have left the models without anything to try and learn how to predict. Then, on the other hand, HTR is a predictor variable that would have a great deal of importance for training the models. HTR is the feature that represents the state of the matchup at half-time and in my personal experience, the result at half-time gives me a good idea of how the game will end. That being said, dropping either of these feature columns was out of the question. This issue was rectified by replacing 'H', 'D', and 'A' with 1, 2, and 3 respectively. In doing so, both the FTR and HTR feature columns were changed from being `characters` to `numeric`. Furthermore, training and testing sets that worked for each of the models had finally been created. To ensure that both models were trained and tested using the identical dataset, the Date, Division, HomeTeam, and AwayTeam feature columns were removed from the training and testing sets used by the random forests model.

Another challenge faced during this thesis dealt with having to remove the FTGH and FTAG feature columns from the training and testing sets for each model. During the initial runs of each model, FTHG and FTAG constantly appeared as the two most important predictors according to both models, while most other features were being ignored. In this case, FTHG and FTAG were both serving as direct proxies by the models for the full-time result (FTR) of each El Clásico matchup in each model's testing set. After removing these features from the testing and training set for each model, the feature importance values were finally being distributed in a much more feasible manner for both models.

3.2 Random Forests Model

Random forests is a popular machine learning method that can for both regression and classification problems. In the thesis, the random forests model is used for classification rather than regression. For classification problems, random forests models build decision trees using different subsamples of the dataset they are given. This technique is what is referred to as bagging. The random forests model then uses the majority vote from these decision trees to classify its predictions. For regression problems, random forests models use the average of the outputs from the decision trees that they build to classify their predictions. [11] One of the main benefits of using random forests is the reduced risk of overfitting. Decision trees alone will usually fit all of the training data samples closely together, which can cause the model to overfit the data. Random forests models can sidestep this issue because when there are many decision trees in a random forest the classifier can prevent overfitting the model by taking the majority vote (or average in the case of regression) of the uncorrelated trees. Doing so lowers the variance and prediction error in the model's results.

The random forests model developed for this thesis was implemented using the `cforest` function from `R`'s `party` package, following Schauberger and Groll (2018). To analyze the model, the dataset being used must be split into a training set and a testing set. The training set will be used to fit and train the model and the testing set will be used to generate the model's predictions. A 75:25 split ratio was put into practice for splitting the dataset being used for the analysis of this model into a training and testing set, respectively. This means that 75% of the dataset was randomly selected and was then used as the training set for the model, while the remaining 25% of the dataset was used as the testing set. When running the `cforest` function, which executes the training phase of the model, it was important for the model to be seeded to ensure the reproducibility of the resulting trained model. The `set.seed` function was used to set a seed number of 123 for training the random forests model. Seeding the split of the dataset for the training and testing sets was also important for the reproducibility of different training and testing datasets. For the initial run of the random forests model, A seed number of 1 was used.

After running the training phase for the random forests model, we are left with a model object that can be used to predict the outcomes for data that the model has not seen before. To test the random forests model I've created, the `predict` function, which is included as a part of the `stats` package in `R`, was used. As a result of running this function on the random forests model, a list of predictions is produced. The list includes the model's predicted FTR for every matchup that is a part of the test set.

3.3 XGBoost Model

Xgboost is another machine learning method that has gained much popularity in recent years. Xgboost is another decision-tree-based model, however, as opposed to bagging trees as random forests does, xgboost works by boosting trees. Boosting is the process of selecting a random sample of data, fitting it to a model, and then training it sequentially over a number of iterations. [12] Here, each model tries to compensate for the weaknesses of the previous model. With each iteration, the weaker prediction algorithms of every individual classifier are combined to form a stronger prediction algorithm, which culminates in the final xgboost model. Similar to random forests, xgboost models can be used for classification and regression problems alike. For this thesis, the xgboost model is for classification. It is important to note that in order to train the model to predict in terms of classification, the `objective` hyperparameter must be set to `multi:softmax`, otherwise the model will use logistic regression by default.

To implement the xgboost model, the `xgboost` package in `R` was employed. Analogous to preparing the training and testing sets for the random forests model, a 75:25 ratio split was used for the dataset. Again, to ensure reproducibility throughout the analysis of the model and to maintain the same training and testing set as the random forest model, `set.seed` was used to set a seed of 1 for the initial splitting of the dataset being used to analyze the xgboost model. Although there was no specific benefit to using the same seed number to train the xgboost model as was used to train the random forests model, a seed number of 123 was set to ensure reproducibility of the trained xgboost model. Additionally, it is important to mention that the training and testing sets have to be represented as a matrix in order for the sets to be accepted into the xgboost model. This can be done by passing the training and testing data sets into the `as.matrix` function, which is included as a part of the `base R` package.

As with the random forests model, the result of training the xgboost model is a model object that can be used to form predictions on new data. Using the same `predict` function used in section 3.2, a list of predictions made by the xgboost model is produced.

4 Results

4.1 Initial Comparison of Model Results

The prediction lists that resulted from using the `predict` function mentioned in sections 3.2 and 3.3 were used to analyze the predictive capability of each model. To be specific, a data frame was created for each of the resulting lists of predictions from the random forests and xgboost models. Then, the FTR feature column from each model's testing set was appended to the data frames housing each model's prediction list respectively. Finally, a new column named Correct was created in each data frame that

served as a flag as to whether or not the model's prediction was correct. If a model's prediction was equal to the FTR for a given observation, then the model's prediction was correct and a 1 is placed into the new Correct feature column for that observation. Otherwise, the model's prediction was wrong and a 0 is placed into the new feature column. To get the final accuracy measurements for each model, the sum of each data frame's Correct column was divided by the length of the column. After comparing the first accuracy measures for both the random forests and xgboost models, it was found that the xgboost model outperformed the random forests model by nearly 30%. While the random forests model was able to predict the outcomes of its testing set with 63.6% effectiveness, the xgboost model was able to predict the outcome of its own testing set with 90.9% accuracy. However, in the next section, it is revealed how these initial measurements are unsound for comparing the predictive capabilities of the two models.

Figures 1 and 2 present the feature importance values associated with each predictor for the random forests and xgboost models respectively. The feature importance of each model was obtained using the `vip` function, which is included in `R`'s `vip` package. There are differences between the level of importance each model assigns to each of its predictors. One of the main differences is that the random forests model can assign negative importance values for its predictors based on how disadvantageous each predictor is during the model's training process. Meanwhile, the xgboost model only assigns positive importance values to its predictors, if any. Another difference between both figures is the top predictors for each model. Although both models seem to agree that HTR, HTHG, HST, and AST are four important predictors during the training phases of each model, it can be seen that the random forests model considers the half-time stats to be the two most important predictors for predicting the outcome of an El Clásico matchup, while the xgboost model considers the shooting statistics to be its two most important predictors. The differences between the importance values for each model can be attributed to the fact that each model undergoes a different training process.

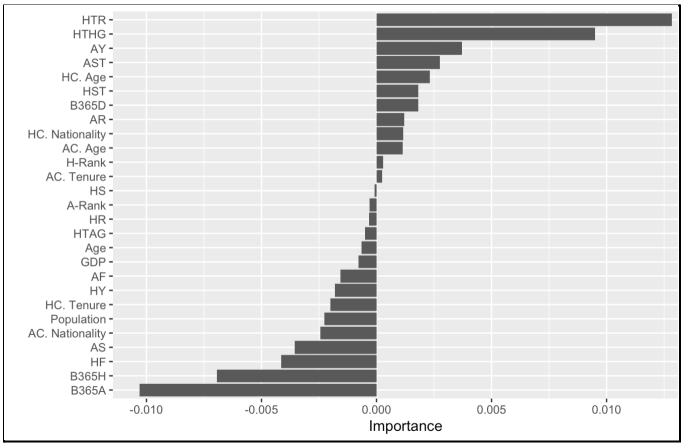

Figure 1 Feature importances for the random forests model

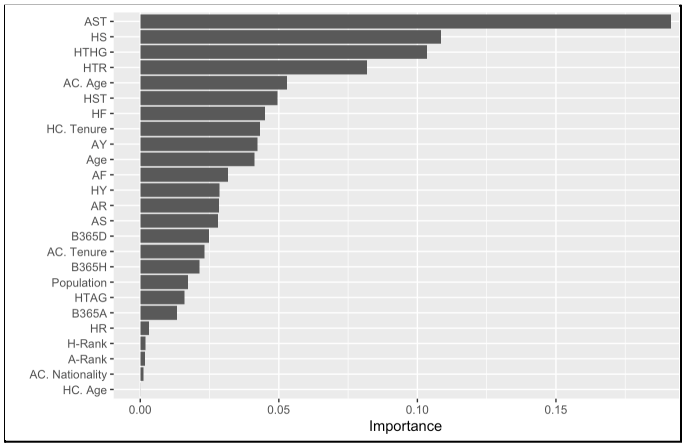

Figure 2 Feature importances for the xgboost model

4.2 Cross-Validation

According to the initial predictions made by the random forests and xgboost model, the later model outperforms the former by nearly 30%. This would indicate that in terms of predictive ability for soccer, the xgboost model is the obvious choice over the random forests model. Furthermore, these measurements suggest that the xgboost model will be capable of predicting the outcome of any number of soccer matchups with approximately 90% accuracy, which would be astounding. However, these initial prediction measurements do not accurately represent the predictive abilities of these models. To get a reliable depiction of the predictive accuracy of the models, cross-validation must be performed. Cross-validation is the process of using different samples of a dataset as training and testing sets to get a generalized scope of a model's predictive ability. [13] Without cross-validation, only information on how a model performs using a single sample of the dataset as the model's training and test sets is made available, which is not a generalization of a model's predictive performance and is, therefore, unreliable information.

The Monte Carlo cross-validation method can be performed on a model by splitting the dataset that is being used to analyze the model into N randomly split training and testing sets. Each respective training and testing set is then used to test the predictive accuracy of the model. After obtaining an accuracy measure from each of the N randomly split training and testing sets, the average of those accuracies is taken and the result is used as the model's predictive accuracy measure. To apply the Monte Carlo cross-validation method to the random forests and xgboost models analyzed in this thesis, the average accuracies produced using 25 different training and testing sets were taken as the final predictive measurements for each model. The final accuracy measures after cross-validating each model were different from what the initial accuracies suggested. The random forests model was predicting the outcomes of El Clásico with an average accuracy of 55.7%, which is 7.9% less accurate than the initial measure. The xgboost

model on the other hand witnessed a dramatic decrease of 25.4%, as the average accuracy was 65.5%. This suggests that the xgboost model is still a better choice than the random forests model in terms of predicting outcomes for El Clásico. Although the conclusion would have been similar in both cases, the results from cross-validating the models demonstrate that using the initial predictive measures for comparing the models would have been unsound.

4.3 Changing the Dataset

This section explores how the predictive performance of both models changes when tested using different dataset conditions. The results for each experiment were obtained using the same Monte Carlo cross-validation method used in section 4.2. The first experiment questioned how the predictive performance for each model would change if only the 10 most important predictors (taken from the figures above) were included as part of each model's training and testing set? After adjusting the datasets such that they only include the 10 most important features according to each model, the average predictive performance for both models dropped. The random forest model's predictive accuracy under these conditions was 53.8%, while the xgboost model's accuracy was 65.1%. These results suggest that changing the dataset in this manner was neither a benefit nor a detriment to the predictive performance of the xgboost model. However, since the random forests model experienced a 1.9% decrease in predictive performance, it is safe to assume that making this change to the dataset is more harmful to the models than it is beneficial.

The second experiment questioned how the predictive capability of each model changes if the format of the features for each model's datasets were changed to follow the format used by Schauberger and Groll (2018)? Specifically, this means that all numeric features will be represented as the difference between the home team's value and the away side's value. An example of this would be that if HS is equal to 3 for a given matchup, and AS is equal to 7, then the resulting feature, S, would be equal to -4. Applying these changes to the datasets causes the predictive performance for both models to drop notably. The random forest model's accuracy dropped to 52.7% while the accuracy of the xgboost model dropped to 56%. The decrease in predictive performance for both models across the two experiments proposes that the original dataset was more useful in terms of training the models than the new datasets.

5 Conclusion

This thesis compared the predictive performance between random forests and xgboost models. These two different predictive machine learning models were compared based on their ability to predict the outcomes of El Clásico matchups. The models were trained and tested using identical training and testing sets, and models were cross-validated using the Monte Carlo cross-validation method. The comparisons

between the two models finally revealed that the xgboost model outperforms the random forests model by nearly 10%, in terms of predicting the outcomes for El Clásico. Later experiments discovered that by changing the dataset for each model to include only the 10 most important features for each model the accuracy for both models dropped by a small margin. The experiments additionally uncovered that changing the format of the features in the original dataset to match the format used by Schauberger and Groll (2018) did more harm than good for each model's predictive performance as well.

Looking ahead, in terms of predicting the outcomes of El Clásico matchups, there still seem to be possibilities for increasing the overall performance of each model. One way of doing so might be to increase the number of matchups used as part of the dataset. This would increase the training and testing sets for each model, which should provide each model with more data to fit itself to and later test against. Another approach would be to fine-tune the hyperparameters of each model, however, this could be a computationally complex task to handle.

6 References

[1] The history of soccer
https://historyofsoccer.info/the_history_of_soccer#The_Ancient_History_of_Soccer
Accessed 13, April 2022

[2] Legalized sports betting in the U.S. doubled in 2021
https://fortune.com/2022/01/24/legalized-sports-betting-in-the-u-s-doubled-in-2021-heres-why-that-will-continue-after-the-greatest-weekend-in-nfl-playoff-history/
Accessed 13, April 2022

[3] Real Madrid
https://www.footballhistory.org/club/real-madrid.html
Accessed 14, April 2022

[4] FC Barcelona
https://www.footballhistory.org/club/barcelona.html
Accessed 14, April 2022

[5] Schauberger, Gunther, and Andreas Groll. "Predicting Matches in International Football Tournaments with Random Forests." *Statistical Modelling*, vol. 18, no. 5-6, 2018, pp. 460–482., https://doi.org/10.1177/1471082x18799934.

[6] Data Files: Spain
https://www.football-data.co.uk/spainm.php
Accessed 8, February 2022

[7] Who Scored?
https://www.whoscored.com/
Accessed 10, March 2022

[8] Flashscore
https://www.flashscore.co.uk/
Accessed 10, March 2022

[9] Footystats
https://footystats.org/
Accessed 11, March 2022

[10] Soccerway
https://us.soccerway.com/matches/2008/12/13/spain/primera-division/futbol-club-barcelona/real-madrid-club-de-futbol/659769/head2head/
Accessed 11, March 2022

[11] Random Forest
https://www.ibm.com/cloud/learn/random-forest
Accessed 19, 2022

[12] Boosting
https://www.ibm.com/cloud/learn/boosting
Accessed 19, 2022

[13] What is Cross-Validation in Machine Learning? Types of Cross-Validation
https://www.mygreatlearning.com/blog/cross-validation/.
Accessed 20, 2022

7 Code Appendix

Appendix A

Importing libraries and creating the initial version of the El Clásico dataset

```r
library(ggplot2)
library(tidyverse)
library(knitr)
library(readr)
library(party)
library(modelr)
library(xgboost)
library(vip)

# All La Liga matchups from 2008 to 2019
df <- read_csv("./data/combined-pre.csv") %>% select(Div, Date, HomeTeam,
AwayTeam, FTHG, FTAG, FTR, HTHG, HTAG, HTR, HS, AS, HST, AST, HF, AF, HY,
AY, HR, AR, B365H, B365D, B365A)

# need to bring in 2019-2020, 2020-2021, and 2021-2022 data separately bc
these seasons did not combine correctly

df1 <- read_csv("./data/2019-2020.csv") %>% select(Div, Date, HomeTeam,
AwayTeam, FTHG, FTAG, FTR, HTHG, HTAG, HTR, HS, AS, HST, AST, HF, AF, HY,
AY, HR, AR, B365H, B365D, B365A)

df2 <- read_csv("./data/2020-2021.csv") %>% select(Div, Date, HomeTeam,
AwayTeam, FTHG, FTAG, FTR, HTHG, HTAG, HTR, HS, AS, HST, AST, HF, AF, HY,
AY, HR, AR, B365H, B365D, B365A)

df3 <- read_csv("./data/2021-2022.csv") %>% select(Div, Date, HomeTeam,
AwayTeam, FTHG, FTAG, FTR, HTHG, HTAG, HTR, HS, AS, HST, AST, HF, AF, HY,
AY, HR, AR, B365H, B365D, B365A)

# All La Liga matchups from the 2008-2009 season up until the 2021-2022
season
df <- rbind(df, df1, df2, df3)
```

```r
# El Clásico 08/09 season to 21/22
classico <- df %>% filter(HomeTeam=="Real Madrid" & AwayTeam=="Barcelona"
| AwayTeam=="Real Madrid" & HomeTeam=="Barcelona")

write_csv(classico, "./data/classico.csv")
```

Appendix B

Creating the final datasets for each model and splitting the datasets into each model's training and testing set

```r
# new dataset with El Clásico matchups from ALL competitions
classico_new <- read_csv("./data/Copy_of_classico.csv")
# xgboost dataset
classico_for_xgb <- classico_new %>% select(-Date, -Div, -FTHG, -FTAG,
-HomeTeam, -AwayTeam)

# random forests dataset
classico_no_chars <- classico_new %>% select(-Date, -Div, -FTHG, -FTAG,
-HomeTeam, -AwayTeam)

# change data type to factor
classico_no_chars[,] <- lapply(classico_no_chars[,], as.factor)
```

```r
# splitting each dataset into training and testing sets
set.seed(1) #seed for reproducibility
index <- round(nrow(classico_no_chars) * 0.25, digits=0)
test_indices <- sample(1:nrow(classico_no_chars), index)
train_set <- classico_no_chars[-test_indices,]  #75% training set
test_set <- classico_no_chars[test_indices,] #25% test set

set.seed(1)
index_xgb <- round(nrow(classico_for_xgb) * 0.25, digits=0)
test_indices_xgb <- sample(1:nrow(classico_for_xgb), index_xgb)
train_set_xgb <- classico_for_xgb[-test_indices_xgb,] #75% training set
test_set_xgb <- classico_for_xgb[test_indices_xgb,] #25% test set
```

Appendix C

Training and creating the random forests model

```r
# training the random forests model
set.seed(123)
cforest_model <- cforest(FTR~.,
                         data = train_set,
                         control = cforest_control(ntree=300,
                                                   mtry=5,
                                                   minsplit = 6,
                                                   minbucket = 2
                                                   )
                         )
```

```
```

Appendix D

Obtaining feature importances, prediction list, and an accuracy measure for the random forests model

```{r}
# feature importance for cfotest
set.seed(1)
vip(cforest_model, num_features = 27)
```
```{r}
# prediciton list for cforest
set.seed(1)
rf_pred_test <- predict(cforest_model, newdata=test_set)
rf_pred_test_list <- as.data.frame(rf_pred_test)
rf_pred_test_list
```
```{r}
# accuracy measures for cforest
rf_test_accuracy <- test_set %>% mutate(rf_pred_test = rf_pred_test)
rf_test_accuracy <- rf_test_accuracy %>% mutate(correct =
ifelse(rf_pred_test == FTR,1,0))
rf_accuracy <- sum(rf_test_accuracy$correct) /
length(rf_test_accuracy$correct)
```

Appendix E

Changing the data type of FTR and HTR for the xgboost model

```{r}
# changing HTR for the test set
test_set_xgb["HTR"][test_set_xgb["HTR"] == "H"] <- "1"
test_set_xgb["HTR"][test_set_xgb["HTR"] == "D"] <- "2"
test_set_xgb["HTR"][test_set_xgb["HTR"] == "A"] <- "3"

# changing FTR for the test set
test_set_xgb["FTR"][test_set_xgb["FTR"] == "H"] <- "1"
test_set_xgb["FTR"][test_set_xgb["FTR"] == "D"] <- "2"
test_set_xgb["FTR"][test_set_xgb["FTR"] == "A"] <- "3"

test_set_xgb[,] <- lapply(test_set_xgb[,], as.numeric)

test_set_xgb_noFTR <- test_set_xgb %>% select(-FTR)

# changing HTR for the train set
train_set_xgb["HTR"][train_set_xgb["HTR"] == "H"] <- "1"
```

```
train_set_xgb["HTR"][train_set_xgb["HTR"] == "D"] <- "2"
train_set_xgb["HTR"][train_set_xgb["HTR"] == "A"] <- "3"

# changing FTR for the train set
train_set_xgb["FTR"][train_set_xgb["FTR"] == "H"] <- "1"
train_set_xgb["FTR"][train_set_xgb["FTR"] == "D"] <- "2"
train_set_xgb["FTR"][train_set_xgb["FTR"] == "A"] <- "3"

train_set_xgb[,] <- lapply(train_set_xgb[,], as.numeric)

train_set_xgb_noFTR <- train_set_xgb %>% select(-FTR)
```

Appendix F

Training and creating the xgboost model

```{r}
# training xgboost model
set.seed(123)
xgb_model <- xgboost(as.matrix(train_set_xgb_noFTR),
                     train_set_xgb$FTR,
                     nthread = 2,
                     nrounds = 15,
                     subsample = 0.5,
                     objective = "multi:softmax",
                     num_class = 4)
```

Appendix G

Obtaining the feature importances, prediction lists, and an accuracy measure for the xgboost model

```{r}
# feature importance for xgboost
set.seed(1)
vip(xgb_model, num_features = 27)
```
```{r}
# prediction list for xgboost
pred_test <- predict(xgb_model, as.matrix(test_set_xgb_noFTR))
train_prediction_list <- as.data.frame(pred_train)
test_prediction_list <- as.data.frame(pred_test)
```
```{r}
# accuracy measure for xgboost
xgb_test_accuracy <- test_set_xgb %>% mutate(pred_test = pred_test)
```

```
xgb_test_accuracy <- xgb_test_accuracy %>% mutate(correct =
ifelse(pred_test == FTR,1,0))

xgb_accuracy <- sum(xgb_test_accuracy$correct) /
length(xgb_test_accuracy$correct)
```

Appendix H

Printing the accuracy measure of each model

```
```{r}
# accuracy measures for each model
rf_accuracy
xgb_accuracy
```
```

Appendix I

An example of both models being cross-validated

```
```{r}
# The average accuracy of using 25 Different train and test sets that
only include the top 10 variables!
# Monte Carlo cross-validation
seed_num <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 23, 24, 25)

list_rf_accuracy <- c(0.8181818, 0.6363636, 0.7272727, 0.3636364,
0.3636364, 0.4545455, 0.5454545, 0.3636364, 0.4545455, 0.4545455,
0.7272727, 0.8181818, 0.3636364, 0.5454545, 0.6363636, 0.3636364,
0.5454545, 0.4545455, 0.7272727, 0.5454545, 0.6363636, 0.3636364,
0.5454545, 0.5454545, 0.4545455)

list_xgb_accuracy <- c(0.8181818, 0.6363636, 0.6363636, 0.5454545,
0.3636364, 0.5454545, 0.8181818, 0.5454545, 0.5454545, 0.7272727,
0.7272727, 0.8181818, 0.4545455, 0.7272727, 0.6363636, 0.7272727,
0.7272727, 0.6363636, 0.8181818, 0.6363636, 0.7272727, 0.6363636,
0.6363636, 0.6363636, 0.5454545)

df_acc <- data.frame(seed_num, list_rf_accuracy, list_xgb_accuracy)
rf_CV_acc <- mean(df_acc$list_rf_accuracy)
xgb_CV_acc <- mean(df_acc$list_xgb_accuracy)
rf_CV_acc
xgb_CV_acc
```
```